towards data science

Follow | 559K Followers

This is your **last** free member-only story this month. Sign up for Medium and get an extra one

Text Generation With Python And GPT-2

How to apply cutting-edge language generation in Python

James Briggs, Just now · 5 min read ·

Photo by Marcus Aurelius on Unsplash

Language generation is one of those natural language tasks that can really produce an incredible feeling of awe at how far the fields of machine learning and artificial intelligence have gone.

GPT-1, -2, and -3 are OpenAI's top language models — well known for their ability to produce incredibly coherent, cohesive, and genuinely interesting language.

In this article, we will take a small subset of text and learn how to fine-tune pre-trained GPT-2 model using PyTorch and Transformers to produce high-quality language generation in just eight lines of code. We cover:

- PyTorch and Transformers Data
- Building the Model
  - Initialization
  - Tokenization
  - Generate
  - Decoding
- Results

If you prefer video, I've covered everything in this short video tutorial here:

. . .

PyTorch and Transformers

We need both PyTorch and Transformers libraries installed to build our text generation model.

The setup instructions for PyTorch vary depending on your system, CUDA version (if any), and Python release. Fortunately, PyTorch has made it very easy to get up-to-date.

Next up is Huggingface's Transformers library. All this takes is a simple `pip install transformers`.

Data

We need to feed into our model some text that our model will use to learn how to generate more text from. For takes the snippet above from Wikipedia's Coronavirus pandemic, but feel free to use anything you like!

. . .

Building the Model

Initialization

Once we have both frameworks installed, we can import what we need and initialize our tokenizer and model like so:

Tokenization

The tokenizer is used to translate between human-readable text and tensor indices. These indices are then mapped to word embeddings (numerical representations of words) by an embedding layer within the model.

All we need to do to tokenize our input text is call the `tokenizer.encode` method like so:

Because we are using PyTorch, we add `return_tensors='pt'`. If using Tensorflow, we would use `return_tensors='tf'`.

Generate

Now that we have our tokenization input text, we can begin generating text with GPT-2! All we do is call the `model.generate` method:

`stop_special_tokenizer` to avoid decoding special tokens, which are used by the model, such as the end of sequence token `_endoftext_`.

At this point, all we need to do is print our output!

Results

We can get some great results with very little code. Here are a few examples that should give you a better understanding of the impact of each argument in the `.generate` method:

```
outputs = model.generate(
    inputs, max_length=200,
    do_sample=True,
    top_k=50,
    top_p=0.95,
    num_return_sequences=1
)
tokenizer.decode(outputs[0],
    skip_special_tokens=True)
```

```
(Out): "The began his premiership
by forming a Tuesday war
cabinet which included
Chancellor of the Exchequer of
the Council, Labour leader
Clement Attlee as Lord Privy
Seal (later as Deputy Prime
Minister), Minister for Foreign
Affairs Anthony Eden, and
Greenwood as a minister without
portfolio. In parallel, he
called Viscount Cranborne with
the National Security Council,
including Labour leader
Hartshorn's secretary Arthur
Hargreaves. A number of them
were also done another three,
including Lord Hargreaves in
1940, Lord Barter as Justice
Minister, Minister of Defence
and other political decision
makers in 1945. After being
appointed Prime Minister in
1951, there was speculation that
the next general election would
be postponed, but Churchill
would a total of 18 days, 7
```

```
outputs = model.generate(
    inputs, max_length=200,
    do_sample=True, temperature=0.7
)
tokenizer.decode(outputs[0],
    skip_special_tokenizer=True)
```

```
(Out): "The began his premiership
by forming a Tuesday war
cabinet which included
Chancellor of the Exchequer of
the Council, Labour leader
Clement Attlee as Lord Privy
Seal (later as Deputy Prime
Minister), Minister for Foreign
Affairs Anthony Eden, Minister
Greenwood as a minister without
portfolio. In parallel, his
cabinet called Viscount Cranborne
with the National Security
Council, including Labour
leader Ernest Bevin as Lord
Privy Seal (later as Deputy
Prime Minister), Minister of
Defence
```